1 Modularity: Community Detection in a Network

Recall the goal of the community detection in a network:

> "to quantify the extent to which the network was composed of distinct communities ... *within* which many edges occurred, and *between* which few edges occurred." [2] (italics added)

The key point in this analysis is not only to find large *high-density* subgraphs, as we did earlier, but to find ones with great *contrast* between the *interior* density of the subgraph, and the density of its *connections* to the rest of the graph.

> Merely the finding that a network contains tightly knit groups at all can convey useful information: if a metabolic network were divided into such groups, for instance, it could provide evidence for a modular view of the networks dynamics, with different groups of nodes performing different functions with some degree of independence. [4]

One might ask if it is necessary or helpful to make such an explicit distinction between the interior and exterior densities, because a *largest* high-density subgraph is likely to have few connections to nodes outside of it, or else it could have been enlarged. But, that contrast is not guaranteed. The *community detection* problem makes that contrast and the goal of finding it, explicit.

1.1 Modularity

> Modularity is considered to be one of the main organizing principles of biological networks. A biological network module consists of a set of elements (e.g., proteins/ reactions) that form a coherent structural subsystem and have a distinct function. [3]

But, how exactly should we define a single community or the collection of communities or subgraphs in a graph? A widely used and studied formalization of the concept of a community comes from the definition of the *modularity* of a graph, first proposed in [4].

1.1.1 Definition of modularity

Here, we will develop a formal definition of *modularity*[1] which was proposed in [4], and is very widely used. This definition is very technical, and it may be hard to see what it means. So, we will develop it slowly.

For an undirected graph $G = (V, E)$ of n nodes, let $d(i)$ denote the *degree* of node i, i.e., the number of edges that touch node i, and let A denote the adjacency matrix for graph G. That is, $A(i, j) = A(j, i) = 1$ if and only if there is an edge in G between nodes i and j; and $A(i, j) = A(j, i) = 0$ otherwise.

Next, let the set $\mathcal{C}$ denote a *partition*[2] of the nodes of G into an unknown number of disjoint subsets. Although we don't know how many subsets there are, we denote that number by k, so the partition $\mathcal{C} = (C_1, C_2, ..., C_k)$. Each of the sets C_p is formally called a class of partition $\mathcal{C}$, but in the context of computing modularity, these are the *modules* or *communities*.

For each pair of nodes, $i < j$, in G, let $\mathcal{C}(i, j)$ be an *indicator variable* which has value 1 if and only if nodes i and j are *together* in the *same* class of $\mathcal{C}$. We don't know which class that might be, but i and j are in it together.

The key definition For a pair of nodes (i, j) in G, we define

$$M(i, j) \equiv A(i, j) - \frac{d(i) \times d(j)}{2|E|}, \tag{1}$$

and we will motivate this definition below.

Next, we define the *modularity of a partition* $\mathcal{C}$ as:

$$Q(\mathcal{C}) \equiv \frac{1}{2|E|} \sum_{i < j:\ i,j \texttt{ are nodes in } G} M(i, j) \times \mathcal{C}(i, j). \tag{2}$$

Finally, we define the *modularity of graph* G as the *maximum* $Q(\mathcal{C})$ over all ways of partitioning the nodes of G. More formally,

[1] When we refer to the word "modularity", we will be talking about the technical definition stated in this section. Elsewhere, as in the above quotes, we use similar words, such as "modular" or "module", which are used in a broader, more colloquial way.

[2] Formally, a *partition* $\mathcal{C}$ of a set S is a division of the elements of S into some number of subsets, called *classes*, so that each element of S is in *exactly* one class of $\mathcal{C}$. It follows trivially that the classes are *disjoint*.

$$Q(G) \equiv \max_{\texttt{partition } \mathcal{C} \textit{ of } V} Q(\mathcal{C}). \tag{3}$$

Notice that the *number* of classes in the partition that determines $Q(\mathcal{C})$ is *not specified* in the problem input. Rather, the number of classes is determined in order to *maximize* the value of $Q(\mathcal{C})$. In contrast, many alternative clustering and partitioning methods require specifying the number of classes in a partition as part of the input of a problem instance. This flexibility is one of the strengths of modularity and part of the explanation for its popularity. Another reason is an efficient *greedy* algorithm [1] that, in practice, is reported to compute a good *approximation* of the modularity of a graph, and a good partition.

1.1.2 What do these definitions mean?

The definition of modularity is not easy to understand! I can only give a partial explanation for it, and I have never seen what I consider to be a compelling explanation.[3] But, some explanation is better than no explanation. Still, this will be a hard section, so if you are mostly interested in how to *compute* modularity using ILP, you can skip to Section **??**.

First, notice that the multiplicative term $\frac{1}{2|E|}$ in (2) is a *constant* number, the same for every partition $\mathcal{C}$, so we can ignore it when trying to find a partition that maximizes $Q(\mathcal{C})$, i.e., one that defines $Q(G)$.

Next, consider a specific partition $\mathcal{C}$ of the nodes of G. Notice that if nodes i and j are *not* in the same class of $\mathcal{C}$, then $\mathcal{C}(i,j)$ has value 0, so the term for node pair (i,j) contributes *nothing* to $Q(\mathcal{C})$. The value of $Q(\mathcal{C})$ is therefore only affected by node pairs (i,j) where i and j are in the *same* class of $\mathcal{C}$. So, $Q(\mathcal{C})$ reflects only what is happening *inside* each of the specified classes. That means that the heart of the definition of modularity is the quantity $M(i,j)$. Where does it come from? What does it mean?

A hand-waving answer Focus on a particular class in $\mathcal{C}$, say C_p, and the nodes in C_p. In G there are some edges between pairs of nodes in C_p, and if C_p was chosen well, there will be a *high-density* of edges between the nodes in C_p. But generally, not all possible edges between nodes in C_p will be in G. For a pair of nodes (i,j) in C_p, what $M(i,j)$ tries to measure is how "surprising" or "abnormal" or "unexpected" it is that edge (i,j)

[3] I have seen several that make incorrect mathematical statements, or don't address the key modeling questions.

is in C_p, or is not in C_p. If there are many edges between nodes in C_p that are "unexpected", then we want the numerical contribution of C_p to $Q(\mathcal{C})$ to be large. Conversely, if most of the edges between nodes in C_p are "expected", then we want that numerical contribution to be small. So, the partition $\mathcal{C}$ that maximizes $Q(\mathcal{C})$ is one that groups nodes into classes (modules) with many "unexpected" edges.

OK, but ... *how* do we actually determine if an edge is *expected*? What is *surprising* or *abnormal* depends on which graphs G is *compared* to. The classic answer is to compare G to some set of *randomly generated* graphs. Then, $M(i,j)$ compares the *reality* (i.e., whether edge (i,j) is actually in G) to what we would expect in these *randomly* generated graphs.

OK, but ... *which* randomly generated graphs? This is where *good modeling* and *art* comes in. One general idea is to specify a set of random graphs which are each *similar* to G in many ways, but are not generated in a way that *biases* the density of its induced subgraphs.[4]

OK, but ... it's too vague. So, let's look at a particular set of random graphs, call it $M(G)$, generated in some well-defined way. Since we will be comparing G to the graphs in $M(G)$, and we want those graphs to be similar to G, we will randomly generate graphs with the *same* number of nodes, n, and the *same* number of edges, $|E|$, as G has. Recall that $d(k)$ denotes the *degree* of node k.

Then, for each node k in G, we define

$$p(k) \equiv d(k)/2|E|.$$

So, $p(k)$ is *proportional* to the degree of node k, divided by the total number of edges in G. In fact, $p(k)$ is exactly *half* that ratio.

Exercise 1.1 *Another way to interpret $p(k)$ is based on the famous fact called*

> **The Handshake Lemma:** *In any graph G, $\sum_{node\ k} d(k) = 2|E|$.*

[4]In general when we want to measure how unexpected it is that a graph G has some particular property, we want to compare G to a set of graphs that are randomly generated to be as similar to G as possible, but making sure that the graph generation process does not bias the frequency that the generated graphs have, or don't have, the property of interest.

Using this lemma, explicitly state the alternative interpretation of $p(k)$.

With those definitions, given G, we generate a random graph with the following

> **Procedure M(G):**
>
> Repeat the following $|E|$ times:
>
> Pick a node randomly from the nodes in G. In particular, each node k is picked with probability $p(k)$. Then, pick a second node randomly from the nodes in G. Again, each node k is picked with probability $p(k)$. Call the first picked node i, and the second node j.
>
> Then, put the edge (i, j) into the random graph being constructed. Note that j might be the same node as i, in which case the edge generated is a self-loop at node i.

Note that for a given pair of nodes (i, j), the number of (i, j) edges that could be generated in an execution of this procedure is anywhere from 0 to $|E|$ (unlikely, but possible). Similarly, although the graph generated will have n nodes and $|E|$ edges, the degree of any node i in the new graph might be different from $d(i)$.

With this procedure, we can precisely define $M(G)$ as the *set* of graphs generated if we run the $M(G)$ procedure an *infinite* number of times.

Exercise 1.2 *Above, we simply used $p(k)$, for each node k, as a* probability*. But, for the set of $p(k)$ values to be properly considered as probabilities, each must be between 0 and 1 (inclusive), and the sum $\sum_k p(k)$* must *be equal to 1. Verify that the $p(k)$ values are proper probabilities.*

Averages Of course we can't actually run the procedure an infinite number of times.[5] But if we run it a thousand-gazillion or more times, the set of graphs created would converge to $M(G)$, and we could use that set of graphs to compare to the actual graph G, determining how surprising or unexpected it is that a particular edge (i, j) is in G, compared graphs in $M(G)$. For that comparison, we count the number of (i, j) edges in each of the randomly generated graphs, and then compute the *average* number of

[5] And, anyway, I don't know what infinity is, which is why I gravitate to *finite* math.

(i, j) edges over all the created graphs. That average will converge to what is called the "expected number" of (i, j) edges.[6] We use $E(i, j)$ to denote that expected number. Since in each execution of Procedure $M(G)$, the number of (i, j) edges created could be more than one, it is possible that $E(i, j)$ will be greater than one.

Recapping, we have defined a set of random graphs $M(G)$, and a specific process for generating many random graphs in $M(G)$, and the concept of the expected number of times, $E(i, j)$, that the edge (i, j) appears in a generated graph. So, $E(i, j)$ could be used to measure how surprising it is that the pair (i, j) is an edge in the given graph G, compared to being in a graph in $M(G)$. Following that viewpoint, we would redefine $M(i, j)$ as

$$A(i, j) - E(i, j),$$

and use those $M(i, j)$ values in the definitions of $Q(\mathcal{C})$ and $Q(G)$. But, actually generating a brazilian graphs in $M(G)$ in order to determine $E(i, j)$ values is wildly impractical.

Fortunately, we can calculate $E(i, j)$ *analytically*, i.e, by a formula, *without* actually having to run procedure $M(G)$ even once. Here is the derivation of the formula for $E(i, j)$.

Deriving $E(i, j)$ Looking at procedure $M(G)$, we see that in a *single* pick of an edge, the probability that the particular edge (i, j) is picked is equal to the probability that node i is chosen as the first node, times the probability that node j is chosen as the second node; plus the probability node j is chosen as the first node, times the probability that node i is chosen as the second node. That probability is exactly:

$$2 \times p(i) \times p(j) =$$

$$\frac{2 \times d(i) \times d(j)}{(2 \times |E|)^2}.$$

[6]The term "expected number" here has a precise, technical meaning in probability theory, and does not have exactly the same meaning as it does in colloquial speech. We won't attempt a formal definition of the term, but the concept of the converging average is sufficiently correct for our purposes.

Now in procedure $M(G)$, there are $|E|$ edges that are picked. The edge picks are independent of one another, so the probability that edge (i, j) is picked is the same on each edge pick. A theorem in probability theory, called the *linearity of expectation*, says that in order to determine $E(i, j)$, we can just multiply the number of edge picks by the probability that edge (i, j) is picked in a single edge pick. This gives,

$$E(i,j) = |E| \times \frac{2 \times d(i) \times d(j)}{(2 \times |E|)^2} =$$

$$\frac{d(i) \times d(j)}{2|E|}.$$

We conclude What we have derived is that if we define $M(i, j)$ as $(A(i, j) - E(i, j))$, we can *calculate* $M(i, j)$ as

$$A(i,j) - \frac{d(i) \times d(j)}{2|E|},$$

which is exactly the term for the node pair (i, j) that appears in (1), the definition of $M(i, j)$. And that is the best I can do to explain the *derivation* of $M(i, j)$.

All of this this should make good intuitive sense. For example, if nodes i and j both have high degrees, it shouldn't be terribly surprising if there is an edge between them, and conversely, it would be somewhat surprising if there was no (i, j) edge. This corresponds well to the definition of $M(i, j)$, because when both nodes have high degree, $\frac{d(i) \times d(j)}{2|E|}$ will be large. So if (i, j) is an edge, $M(i, j)$ will be a small number, and when (i, j) is not an edge, $M(i, j)$ will be even smaller.

Exercise 1.3 *Thinking through the case when both nodes i and j have high degree, as above, to see that the formal definitions and derivations agree with intuition, is something that I call a "sanity check". Do a sanity check for the case that both nodes have low degrees.*

Is $M(G)$ the best class to compare with? We said that we want to compare G to graphs in a class of random graphs that are similar to G, but are generated without biasing the density of subgraphs. The procedure that defines $M(G)$ does produce graphs with the same number of nodes and edges as G, and without any explicit consideration for the distribution of subgraph densities. Moreover, if we define $Ed(i)$ as the *expected degree* of node i, over the graphs generated in $M(G)$, $Ed(i)$ is $2 \times |E|$ times the probability of picking node i in any node pick, which is

$$2 \times |E| \times \frac{d(i)}{2 \times |E|} = d(i).$$

So, the set of graphs generated are similar to G in this way also. But is that good enough so that the given definition of modularity is useful? That requires an empirical answer.

Actually, we might be able to do better. It is easy to create a procedure[7] that generates random graphs where in each graph, the degree of any node is exactly its degree in G. That is, the degree of a node, i, will be $d(i)$ in *every* graph generated, although the number of copies of an edge (i, j) can vary. So, those graphs are even *more* similar to G than are the graphs in $M(G)$, and yet they are still randomly generated. Why don't we compare G to that set of graphs instead of to $M(G)$?

I think the answer is that we (or at least I) don't know an analytical way to compute $E(i, j)$ over that set of graphs. If we wanted to know those $E(i, j)$ values, we would have to actually run the generation procedure a gazillion times. So, the use of the definition of $M(i, j)$ given in (1) may be the result of a compromise between the best modeling, and what is computationally more efficient. And that is the best I can do to explain the meaning of $M(i, j)$.

Whew!! OK, this is not easy going. If you don't yet have a feel for $M(i, j)$, $Q(\mathcal{C})$ or $Q(G)$, just consider $Q(\mathcal{C})$ to be a value that can be computed, given $\mathcal{C}$; and $Q(G)$ to be a value that can also be computed, in

[7]The basic idea is as follows: Start with G and cut each edge in the middle, so that each edge becomes two *stubs*. The number of stubs at this point is exactly $2|E|$, since there were $|E|$ edges, and each one becomes two stubs. Then if we *randomly* pair up the unattached ends of the stubs to form $|E|$ edges, the result will be a graph G' where each node has the same degree in G' as in G, since the attached end of each stub didn't change. It is possible that G' would be graph G again, but there are many other possibilities as well. This class of graphs allows parallel edges and self-loops.

principle, given G; and realize that many people who work on the community detection problem think these values have meaning for that problem.

References

[1] V. Blondel, J.L Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10:10008–10020, 2008.

[2] L. Elbroch, M. Levy, M. Lubell, H. Quigley, and A. Caragiulo. Adaptive social strategies in a solitary carnivore. *Science Advances*, 3: e1701218, 2017.

[3] A. Kreimer, E. Borenstein, U. Gophna, and E. Ruppin. The evolution of modularity in bacterial metabolic networks. *Proceedings of the National Academy of Sciences (USA)*, pages 6976–6981, 2008.

[4] M. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences (USA)*, 103:8577–8582, 2006.